About the Termination Detection in the Asynchronous Message Passing Model (Extended Abstract)

Jérémie Chalopin[1], Emmanuel Godard[2], Yves Métivier[1] and Gerard Tel[3]

[1] LaBRI UMR 5800
Université Bordeaux 1, ENSEIRB,
351 cours de la Libération
33405 Talence, France
{chalopin,metivier}@labri.fr
[2] LIF UMR 6166
Université de Provence
39 rue Joliot-Curie
13453 Marseille France
{egodard@cmi.univ-mrs.fr}
[3] Department of Computer Science
University of Utrecht
P.O. box 80.089, 3508 TB Utrecht
The Netherlands
{gerard@cs.uu.nl}

Abstract. The paper presents a complete characterisation of the families of networks in which distributed computations can be performed in a process terminating manner, that is, with *explicit termination* in the asynchronous message passing model. The characterisation encompasses all criteria that have been formulated in the past that were known to influence explicit termination: topological restriction (tree or complete networks), topological knowledge (size or diameter), and local knowledge to distinguish nodes (identities or a leader). These results are now presented as corollaries of a single generalising theorem. In addition our characterisation covers combinations of these, as well as new criteria.

1 Introduction

Starting with the works by Angluin [Ang80] and Itai and Rodeh [IR81], many papers have discussed the question what functions can be computed by distributed algorithms in networks where knowledge about the network topology is limited.

Two important factors limiting the computational power of distributed systems are *symmetry* and *explicit termination*, and both have been found to be connected with the graph-theoretic concept of coverings. Impossibility proofs for distributed computations quite often use the *replay* technique. Starting from a (supposedly correct) execution of an algorithm, an execution is constructed in which the same steps are taken by nodes in a different network. The mechanics of distributed execution dictate that this can happen, if the nodes are *locally* in the same situation, and this is precisely what is expressed by the existence of coverings.

Some functions can be computed by an algorithm that terminates *implicitly* but not by an *explicitly* terminating algorithm. In an implicitly terminating algorithm, each execution is finite and in the last state of the execution each node has the correct result. However, the nodes are not aware that their state is the last one in the execution. The impossibility result implies that such awareness can never be obtained in a finite computation.

During the nineteen eighties there were many proposals for *termination detection* algorithms: such algorithms transform implicitly into explicitly terminating algorithms. As it is explained in [Mat87], they superimposed on a given so-called basic computation a control computation which enables one or more of the processes to detect when the termination condition holds for the basic computation.

It is not easy to detect whether a distributed algorithm has reached a state where no process is active and no message is in transit. Several conditions were found to allow such algorithms and for each of these conditions a specific algorithm was given (see [Tel00] Chap. 8 and [Mat87]). These conditions include: a unique *leader* exists in the network [Ang80], the network is known to be a tree [Ang80], a bound of the diameter of the network is known [SSP85], the nodes have different identification numbers.

1.1 The Main Result

In this paper we show that these four conditions are just special cases of one common criteria, namely that the local knowledge of nodes prohibits the existance of quasi-coverings of unbounded depth. Moreover, we generalise the algorithm by Szymanski *et al.* [SSP85] to a common algorithm that works in all graph families without quasi-coverings of unbounded depth. It is explained in Lemma 14. We also prove, by generalising the existing impossibility proofs to the limit, that in families with quasi-coverings of unbounded depth, termination detection is impossible. Thus, the generalised algorithm can be considered as an *universal termination detection algorithm* that can be applied in all cases where detection is possible at all. It is precisely what is stated in Theorem 22.

From this theorem and [CM05] we deduce a characterisation of families of labelled graphs which admit an election algorithm: Theorem 23.

1.2 Related Works

For the asynchronous message passing model characterisations of graphs permitting a leader election algorithm, a spanning tree construction algorithm and a topology recognition algorithm have been obtained by [YK96]. For this, they introduced the concept of view. The view from a vertex v of a graph G is an infinite labelled rooted tree obtained by considering all labelled walks in G starting from v. The characterizations use also the notion of symmetricity. The symmetricity of a graph depends on the number of vertices that have the same view. Algorithms developped in our work are totally asynchronous; in [YK96] algorithms need a pseudo-synchronisation. Consider a graph G with n vertices and m edges, in Yamashita and Kameda algorithms the size of each message can be 2^n whereas in our algorithm the size is bounded by $O(m \log n)$.

In [BV99,BV01], Boldi and Vigna study a model where a network is represented by a directed graph. In one computation step, a process can modify its state according to the states of its in-neighbours. In [BV01], they use fibrations to characterize the tasks that can be computed in an anonymous network, provided a bound on the network is known. In [BV99], they give a characterization of what can be computed with arbitrary knowledge; their results are based on the notion of view that is adapted from the work of Yamashita and Kameda [YK96]. From our results, if a task can be computed on a network, provided a bound on the size is known, then we can also detect the termination of the algorithm: in some sense, we generalize the results presented in [BV01]. On the other hand, when a bound on the size is not available, there exist some tasks that are computable in the sense of [BV99] but there does not exists any algorithm that enables to detect that the computation is globally over. In [MT00] a characterisation of networks which autorise explicit termination has been given in the local computation model where in a step a vertex can read and write its state and the states of adjacent vertices.

1.3 The Tools

Coverings and Quasi-coverings. Distributed tasks like election, enumeration (assigning different numbers to the nodes), and mutual exclusion require the network to reach a *non-symmetric* state. A network state is symmetric if it contains different nodes that are in exactly the same situation; not only their local states, but also the states of their neighbors, of their neighbors' neighbors, etcaetera. That is, there exists a "local similarity" between different nodes of infinite radius.

The replay argument shows that different nodes that are locally similar with infinite radius will exhibit the same behavior in some infinite computation. Thus, there is no algorithm that guarantees that the symmetry ceases in all finite computations. Symmetry can be broken only by randomized protocols.

It is not difficult to see that local similarity of infinite radius may exist in finite graphs. The classical example is a ring G_6 of six nodes, with initial states a, b, c, a, b, c. Indeed, the two nodes with state a both have neighbors in state b and c, and so on, so the local similarity exists over an infinite radius.

The ring G_6 can be mapped into a ring G_3 with only three nodes, with initial states a, b, and c, in such a way that each node is mapped to a node with the same initial state *and with the same states in neighbors.* Such a mapping is called a *covering* and is the mathematical tool to prove the existence of symmetries.

Networks in which symmetries exist were called *ambiguous* by Mazurkiewicz [Maz97]; impossibility of symmetry breaking can be shown for these *graphs.* Termination detection requires that a node certifies, in a *finite* computation, that all nodes of the network have completed their computation. However, in a finite computation only information about a bounded region in the network can be gathered. The algorithm by Szymanski, Shy, and Prywes does this for a region of pre-specified diameter; the assumption is necessary that the diameter of the *entire network* is known. This implies, that termination detection, unlike symmetry breaking, is possible in every *graph*, but: provided some *knowledge.*

Network knowledge in an algorithm is modeled by a graph *family* in which the algorithm is required to work. The detection algorithm by Szymanski *et al.* can be generalized in this way to work in a graph family $\mathcal{F}$. Nodes observe their neighborhood and determine in what graph H of $\mathcal{F}$ they are. Then they try to get a bound k on the radius to which a different graph of $\mathcal{F}$ can be locally similar to H, and then certify that all nodes within distance k are completed. The universal termination detection algorithm thus combines the universal graph reconstruction with (minimal) topological knowledge [Maz97] and a known termination detection algorithm. Of course the approach fails if a graph $H \in \mathcal{F}$ is locally similar, with *unbounded* radius, to other graphs in $\mathcal{F}$. Local similarities of this type are made precise in the notion of *quasi-coverings*. Fortunately, the impossibility proofs for termination detection can be extended to cover exactly those families of graphs that contain such unbounded-radius coverings. Consequently, the sketched universal termination detection algorithm is the most general algorithm possible.

Labelled (Di)Graphs and Local Computations on Arcs. We use a labelled directed graph which encodes the network in which processes communicate by asynchronous message passing with a port numbering. The labelling may encode anonymous networks (all the vertices have the same label) or any initial process knowledge. Examples of such knowledge include: (a bound on) the number of processes, (a bound on) the diameter of the graph, the topology, identity or partial identity, distinguished vertices, the sense of direction. The basic events (send, receive, internal, transmission) are encoded by local computations on arcs. From this directed graph, we deduce necessary conditions for the existence of a transformation of algorithms into algorithms having an explicit termination. The conditions are also sufficient (Theorem 22): we give an explicit transformation.

1.4 Overview

The structure of this paper is as follows. Section 2 reviews the definitions of coverings, quasi-coverings and local computations on arcs. Sections 3 explains how to encode a network into an equivalent digraph and basic instructions of the asynchronous message passing model into local computations on arcs. Sections 4 and 5 present a Mazurkiewicz-like algorithm which enables the computation of the maximal knowledge common to all vertices of the network. Section 6 presents the main result and some applications.

2 Preliminaries

The notations used here are essentially standard. Definitions and main properties are presented in [BvL86,Bod89,BV02].

Undirected Graphs, Directed Graphs and Labelled (Di)Graphs. We consider finite, undirected, connected graphs having possibly self-loops and multiple edges, $G = (V(G), E(G), \mathrm{Ends})$, where $V(G)$ denotes the set of vertices, $E(G)$ denotes the set of edges and Ends is a map assigning to every edge two vertices: its ends. Two vertices u and v are said to be adjacent or neighbours if there exists an edge e such that $\mathrm{Ends}(e) = \{u, v\}$. A simple graph $G = (V(G), E(G))$ is a graph without self-loop or multiple edges. For an edge e, if the vertex v belongs to $\mathrm{Ends}(e)$ then we say that e is incident to v. The set of neighbours of v in G, denoted $N_G(v)$, is the set of all vertices of G adjacent to v. A homomorphism between G and H is a mapping $\gamma\colon V(G) \cup E(G) \to V(H) \cup E(H)$ such that if e is an edge of G and $\mathrm{Ends}(e) = \{u, v\}$ then $\mathrm{Ends}(\gamma(e)) = \{\gamma(u), \gamma(v)\}$. We say that γ is an isomorphism if γ is bijective and γ^{-1} is a homomorphism, too. We write $G \simeq G'$ whenever G and G' are isomorphic. In some applications we need a direction on each edge of a graph; a graph augmented in this way is called a directed graph or a digraph. More formally, a digraph $D = (V(D), A(D), s_D, t_D)$ is defined by a set $V(D)$ of nodes, a set $A(D)$ of arcs and by two maps s_D and t_D that assign to each arc two elements of $V(D)$: a source and a target (in general, the subscripts will be omitted); if a is an arc, $\mathrm{Ends}(a)$ denotes the set $\{s(a), t(a)\}$; the arc a is said to be going out of $s(a)$ and coming into $t(a)$. A self-loop is an arc with the same source and target.

A symmetric digraph (V, A, s, t) is a digraph endowed with a symmetry, that is, an involution $Sym : A \to A$ such that for every $a \in A : s(a) = t(Sym(a))$. A digraph D is strongly connected if for all vertices v_1 and v_2 there is a directed path from v_1 to v_2. A digraph homomorphism γ between the digraph D and the digraph D' is a mapping $\gamma\colon V(D) \cup A(D) \to V(D') \cup A(D')$ such that if u, v are vertices of D and e is an arc such that $u = s(e)$ and $v = t(e)$ then $\gamma(u) = s(\gamma(e))$ and $\gamma(v) = t(\gamma(e))$. Let $G = (V, E)$ be a simple graph.

Throughout the paper we will consider graphs where vertices and edges are labelled with labels from a recursive label set L. A graph G labelled over L will be denoted by (G, λ), where $\lambda\colon V(G) \cup E(G) \to L$ is the labelling function. The graph G is called the underlying graph and the mapping λ is a labelling of G. Let G be a graph, let L be a set of labels, and let *neutral* be a label that is not in L. A partial labelling λ of G with labels from L defined on a subset B of $V(G) \cup E(G)$ is canonically extended to $V(G) \cup E(G)$ by putting for each $x \in ((V(G) \cup E(G)) \setminus B)$: $\lambda(x) = neutral$.

A mapping $\gamma\colon V(G) \cup E(G) \to V(G') \cup E(G')$ is a homomorphism from (G, λ) to (G', λ') if γ is a graph homomorphism from G to G' which preserves the labelling, i.e., such that $\lambda'(\gamma(x)) = \lambda(x)$ for every $x \in V(G) \cup E(G)$. In some applications we need several labelling functions for a given graph G. Let $(\lambda_1, ..., \lambda_k)$ be a tuple of

labelling functions of G, the labelled graph obtained with this tuple is denoted $(G, (\lambda_1, ..., \lambda_k))$, the label of an element $x \in V(G) \cup E(G)$ is $(\lambda_1(x), ..., \lambda_k(x))$. Labelled graphs will be designated by bold letters like $\mathbf{G}$, $\mathbf{H}$, ... If $\mathbf{G}$ is a labelled graph, then G denotes the underlying graph. The same definitions are available for digraphs.
Fibration, covering and quasi-covering. Coverings and quasi-coverings are fundamental tools in this work.

Definition 1. *A fibration between the digraphs D and D' is a homomorphism φ from D to D' such that for each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = t(a')$ there exists a unique arc a in $A(D)$ such that $t(a) = v$ and $\varphi(a) = a'$.*

The arc a is called the lifting of a' at v, D is called the total digraph and D' the base of φ. We shall also say that D is fibred (over D'). The fibre over a vertex v of D' is the set $\varphi^{-1}(v)$ of vertices of D. A fibre over v is trivial if it is a singleton, i.e., $|\varphi^{-1}(v)| = 1$. A fibration is nontrivial if at least one fibre is nontrivial, trivial otherwise; it is proper if all fibres are not trivial. A graph D is fibration prime if it cannot be fibred non trivially, that is, every surjective fibration is an isomorphism. In the sequel directed graphs are always strongly connected and total digraphs non empty thus fibrations will be always surjective.

Definition 2. *An opfibration between the digraphs D and D' is a homomorphism φ from D to D' such that for each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = s(a')$ there exists a unique arc a in $A(D)$ such that $s(a) = v$ and $\varphi(a) = a'$. A covering projection is a fibration that is also an opfibration.*

If a covering projection $\varphi : D \to D'$ exists, D is said to be a covering of D' via φ. Covering projections satisfy:

Proposition 3. *A covering projection $\varphi : D \to D'$ with a connected base and a nonempty covering is surjective; moreover, all the fibres have the same cardinality. This cardinality is called the number of sheets of the covering.*

As for fibrations, a digraph D is covering prime if there is no digraph D' not isomorphic to D such that D is a covering of D' (i.e., D is a covering of D' implies that $D \simeq D'$). Let D and D' be two digraphs such that D is a surjective covering of D' via φ. If D' has no self-loop then for each arc $a \in A(D) : \varphi(s(a)) \neq \varphi(t(a))$. Finally the following property is a direct consequence of the definitions and it is fundamental in the sequel of this paper :

Proposition 4. *Let D and D' be two digraphs such that D' has no self-loop and D is a surjective covering of D' via φ. If $a_1 \neq a_2$ and $\varphi(a_1) = \varphi(a_2)$ then $Ends(a_1) \cap Ends(a_2) = \emptyset$.*

The notions of fibrations and of coverings extend to labelled digraphs in an obvious way: the homomorphisms must preserve the labelling.
The last notion we will use is a generalisation of coverings, it is called quasi-coverings.

Definition 5. *Let $\mathbf{D}, \mathbf{D'}$ be two labelled digraphs and let γ be a partial function on $V(\mathbf{D})$ that assigns to each element of a subset of $V(\mathbf{D})$ exactly one element of $V(\mathbf{D'})$. Then $\mathbf{D}$ is a* quasi-covering *of $\mathbf{D'}$ via γ of radius r if there exists a finite or infinite covering $\mathbf{D_0}$ of $\mathbf{D'}$ via δ, vertices $z_0 \in V(D_0)$, $z \in V(D)$ such that:*

1. *$B_{\mathbf{D}}(z, r)$ is isomorphic via φ to $B_{\mathbf{D_0}}(z_0, r)$,*
2. *the domain of definition of γ contains $B_D(z, r)$, and*
3. *$\gamma = \delta \circ \varphi$ when restricted to $V(B_D(z, r))$.*

$card(V(B_D(z, r)))$ is called the size *of the quasi-covering, and z the* center*. The digraph $\mathbf{D_0}$ is called the* associated covering *of the quasi-covering.*

Local Computations on Arcs. In this paper we consider labelled digraphs and we assume that local computations modify only labels of vertices. Digraph relabelling systems on arcs and more generally local computations on arcs satisfy the following constraints, that arise naturally when describing distributed computations with decentralized control:

($C1$) they do not change the underlying digraph but only the labelling of vertices, the final labelling being the result of the computation (*relabelling relations*),
($C2$) they are *local*, that is, each relabelling step changes only the label of the source and the label of the target of an arc,
($C3$) they are *locally generated*, that is, the applicability of a relabelling rule on an arc only depends on the label of the arc, the labels of the source and of the target (locally generated relabelling relation).

The relabelling is performed until no more transformation is possible, i.e., until a normal form is obtained. Let $\mathcal{R}$ be a locally generated relabelling relation, we assume that it is closed under isomorphism, i.e., if $\mathbf{D} \; \mathcal{R} \; \mathbf{D_1}$ and $\mathbf{D'} \simeq \mathbf{D}$ then $\mathbf{D'} \; \mathcal{R} \; \mathbf{D'_1}$ for some labelled digraph $\mathbf{D'_1} \simeq \mathbf{D_1}$. In the remainder of the paper $\mathcal{R}^*$ stands for the reflexive-transitive closure of $\mathcal{R}$. The labelled digraph $\mathbf{D}$ is *$\mathcal{R}$-irreducible* (or just irreducible if $\mathcal{R}$ is fixed) if there is no $\mathbf{D_1}$ such that $\mathbf{D} \; \mathcal{R} \; \mathbf{D_1}$. The relation $\mathcal{R}$ is said noetherian if there is no infinite relabelling chain.

3 From Asynchronous Message Passing to Local Computations on Arcs

The model. Our model follows standard models for distributed systems given in [AW04,Tel00]. The communication model is a point-to-point communication network which is represented as a simple connected undirected graph where vertices represent processes and two vertices are linked by an edge if the corresponding processes have a direct communication link. Processes communicate by message passing, and each process knows from which channel it receives a message or it sends a message. An edge between two vertices v_1 and v_2 represents a channel connecting a port i of v_1 to a port j of v_2. Let ν be the port numbering function, we assume that for each vertex u and each adjacent vertex v, $\nu_u(v)$ is a unique integer belonging to $[1, deg(u)]$. We consider the asynchronous message passing model: processes cannot access a global clock and a message sent from a process to a neighbour arrives within some finite but unpredictable time.

From Undirected Labelled Graphs to Labelled Digraphs. The construction presented in this section may appear technical nevertheless the intuition is very natural and simple, and it is illustrated in Figure 1. A first approximation of a network, with knowledge about the structure of the underlying graph, is a simple labelled graph $\mathbf{G} = (V(\mathbf{G}), E(\mathbf{G}))$. We associate to this undirected labelled graph a labelled digraph $\overleftrightarrow{\mathbf{G}} = (V(\overleftrightarrow{\mathbf{G}}), A(\overleftrightarrow{\mathbf{G}}))$ defined in the following way.

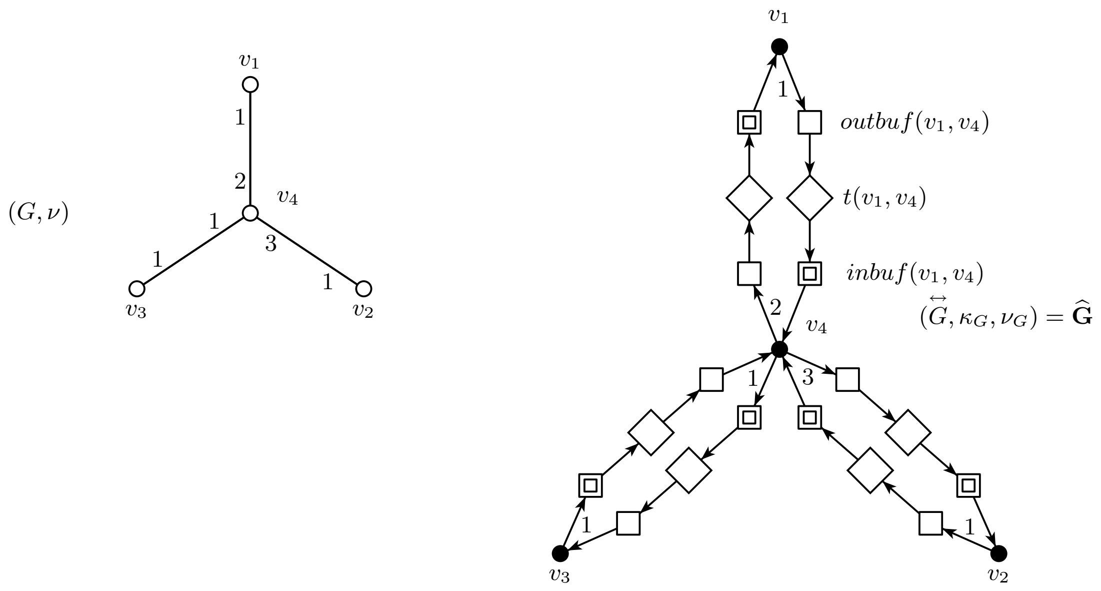

Fig. 1. We adopt the following notation conventions for vertices of $(\overleftrightarrow{G}, \kappa_G, \nu_G)$. A black-circle vertex corresponds to the label **process**, a square vertex corresponds to the label **send**, a diamond vertex corresponds to the label **transmission**, and a double-square vertex corresponds to the label **receive**.

Let u and v be two adjacent vertices of $\mathbf{G}$. We associate to the edge $\{u, v\}$ the set $V_{\{u,v\}}$ of 6 vertices denoted : $\{outbuf(u, v), t(u, v), inbuf(u, v), outbuf(v, u), t(v, u), inbuf(v, u)\}$, and the set $A_{\{u,v\}}$ of 8 arcs equals to: $\{(u, outbuf(u, v)), (outbuf(u, v), t(u, v)), (t(u, v), inbuf(u, v)),$
$(inbuf(u, v), v), (v, outbuf(v, u)), (outbuf(v, u), t(v, u)), (t(v, u), inbuf(v, u)), (inbuf(v, u), u)\}$.

Finally, $V(\overleftrightarrow{\mathbf{G}}) = V(G) \cup (\bigcup_{\{u,v\} \in E(G)} V_{\{u,v\}})$ and $A(\overleftrightarrow{\mathbf{G}}) = \bigcup_{\{u,v\} \in E(G)} A_{\{u,v\}}$.

The arc $(u, outbuf(u, v))$ is denoted $out(u, v)$, $receiver(out(u, v))$ is the vertex v, and the arc $(inbuf(v, u), u)$ is denoted by $in(v, u)$.

If $\mathbf{G} = (G, \lambda)$ then $\overleftrightarrow{\mathbf{G}} = (\overleftrightarrow{G}, \lambda_{\overleftrightarrow{G}})$ where $\lambda_{\overleftrightarrow{G}}(v) = \lambda(v)$ for each $v \in V(G)$.

We need to memorize the meaning (semantic) of vertices thus we label vertices of $\overleftrightarrow{\mathbf{G}}$ with a labelling function κ, the set of labels is: $\{$**process**, **send**, **receive**, **transmission**$\}$,
- if a vertex x of $V(\overleftrightarrow{\mathbf{G}})$ corresponds to a vertex u of $V(G)$ then $\kappa(x) =$ **process**,
- if a vertex x of $V(\overleftrightarrow{\mathbf{G}})$ corresponds to a vertex of the form $outbuf(u, v)$ then $\kappa(x) =$ **send**,
- if a vertex x of $V(\overleftrightarrow{\mathbf{G}})$ corresponds to a vertex of the form $inbuf(u, v)$ then $\kappa(x) =$ **receive**,
- if a vertex x of $V(\overleftrightarrow{\mathbf{G}})$ corresponds to a vertex of the form $t(u, v)$ then $\kappa(x) =$ **transmission**. Using the label *neutral*, κ is extended to $(V(\overleftrightarrow{\mathbf{G}}), A(\overleftrightarrow{\mathbf{G}}))$.

Two adjacent vertices of $(\overleftrightarrow{\mathbf{G}}, \kappa)$ have different labels thus if the digraph $(\overleftrightarrow{\mathbf{G}}, \kappa)$ is a covering of a digraph $\mathbf{D}$ then $\mathbf{D}$ has no self-loop.

Port Numbering and Symmetric Port Numbering. We can notice, that for a digraph $(\overleftrightarrow{\mathbf{G}}, \kappa)$, if we consider a vertex x labelled **process** then $deg^+(x) = deg^-(x)$. Each process knows from which channel it receives a message or it sends a message, that is, each process assigns numbers to its ports. Thus we consider the labelling ν of the arcs of $(\overleftrightarrow{\mathbf{G}}, \kappa)$ coming into or going out of vertices labelled **process** such that for each vertex x labelled **process** the restriction of ν assigns to each outgoing arc a unique integer of $[1, deg^+(x)]$ and assigns to each arc coming into a unique integer of $[1, deg^-(x)]$, such a labelling is a local enumeration of arcs incident to **process** vertices (it corresponds to the port numbering). This enumeration is symmetric, i.e., ν verifies for each arc of the form $out(u, v) : \nu(out(u, v)) = \nu(in(v, u))$; this condition is called the symmetry of the port numbering (or equivalently of ν). Such a port numbering is said symmetric. As usual, using the special label *neutral*, ν is considered as a labelling function of $(\overleftrightarrow{\mathbf{G}}, \kappa)$. The hypothesis of the symmetry of the port numbering is done in [YK96] and it corresponds to the complete port awareness model in [BCG+96]. In the sequel, $(\overleftrightarrow{\mathbf{G}}, \kappa, \nu)$ is denoted by $\widehat{\mathbf{G}}$.

Basic Instructions. As in [YK96] (see also [Tel00] pp. 45-46), we assume that each process, depending on its state, either changes its state, or receives a message via a port or sends a message via a port. Let $Inst$ be this set of instructions. This model is equivalent to the model of local computations on arcs with respect to the initial labelling as it is depicted in the following remark.

Remark 6. Let $\mathbf{G}$ be a labelled graph equipped with a port numbering ν, let $\widehat{\mathbf{G}}$ be the labelled digraph obtained from $\mathbf{G}$, this labelled digraph enables to encode the following events using local computations on arcs:

– an internal event "a process changes its state" can be encoded by a relabelling rule concerning a vertex labelled **process**,
– a send event "the process x sends a message via the port i" can be encoded by a relabelling rule concerning an arc of the form (x, y) with $\kappa(x) = \textbf{process}$, $\kappa(y) = \textbf{send}$ and $\nu((x, y)) = i$,
– a receive event "the process y receives a message via the port i" can be encoded by a relabelling rule concerning an arc of the form (x, y) with $\kappa(x) = \textbf{receive}$, $\kappa(y) = \textbf{process}$ and $\nu((x, y)) = i$,
– an event concerning the transmission control can be encoded by a relabelling rule concerning an arc of the form (x, y) or (y, z) with $\kappa(x) = \textbf{send}$, $\kappa(y) = \textbf{transmission}$ and $\kappa(z) = \textbf{receive}$.

Remark 7. ¿From now one will speak indistinctly of distributed algorithm encoded in the asynchronous message passing model on the labelled graph $\mathbf{G}$ equipped with a port numbering ν or of a distributed algorithm encoded using local computations on arcs on the labelled digraph $\widehat{\mathbf{G}}$.

4 A Mazurkiewicz-like Algorithm

In this section, we recall the algorithm $\mathcal{M}$ inspired by [Maz97] and described in [CM05] and we prove that we can interpret the mailbox of a vertex v at a step i of the computation as a graph $\widehat{\mathbf{H}_i}$ such that $\widehat{\mathbf{G}}$ is a quasi-covering of $\widehat{\mathbf{H}_i}$. Furthermore when the algorithm has reached the final labelling all the vertices compute the same graph $\widehat{\mathbf{H}}$ and $\widehat{\mathbf{G}}$ is a covering of $\widehat{\mathbf{H}}$.

Description of $\mathcal{M}$. We first give a general description of the algorithm $\mathcal{M}$ applied to a labelled graph $\mathbf{G}$ equipped with a port numbering ν. We assume that $\mathbf{G}$ is connected. Let $\mathbf{G} = (G, \lambda)$ and consider a vertex v_0 of G, and the set $\{v_1, ..., v_d\}$ of neighbours of v_0. During the computation, each vertex v_0 will be labelled by a pair of the form $(\lambda(v_0), c(v_0))$, where $c(v_0)$ is a triple $(n(v_0), N(v_0), M(v_0))$ representing the following information obtained during the computation (formal definitions are given below): $n(v_0) \in \mathbb{N}$ is the *number* of the vertex v_0 computed by the algorithm, $N(v_0) \in \mathcal{N}$ is the *local view* of v_0, this view can be either empty or it is a set of the form: $\{((n(v_i), p_{s,i}, p_{r,i}), \lambda(v_i)) | 1 \leq i \leq d\}$, $M(v_0) \subseteq L \times \mathbb{N} \times \mathcal{N}$ is the *mailbox* of v_0 containing the whole information received by v_0 at previous computation steps. Let $(((n(v_i), p_{s,i}, p_{r,i}), \lambda(v_i)) 1 \leq i \leq d)$ be the local view of v_0. For each i, $(n(v_i), p_{s,i}, p_{r,i}))$ encodes a neighbour v_i of v_0, where: $n(v_i)$ is the number of v_i, v_i has sent its number to v_0 via the port $p_{s,i}$, and v_0 has received this message via the port $p_{r,i}$. Each vertex v gets information from its neighbours via messages and then attempts to calculate its own number $n(v)$, which will be an integer between 1 and $|V(G)|$. If a vertex v discovers the existence of another vertex u with the same number, then it compares its own label and its own local view with the label and the local view of u. If the label of u or the local view of u is "stronger", then v chooses another number. Each new number, with its local view, is broadcasted again over the network. At the end of the computation, it is not guaranteed that every vertex has a unique number, unless the graph is covering prime. However, all vertices with the same number will have the same label and the same local view.

Algorithm 1: The algorithm $\mathcal{M}$.

```
Var : $n(v_0)$ : integer init 0 ;
      $N(v_0)$ : set of local view init $\emptyset$;
      $N$ : set of local view ;
      $M(v_0)$ : mailbox init $\emptyset$;
      $M, M_a$ : mailbox;
      $\lambda(v_0), c_a, l$ : element of $L$;
      $i, x, p, q, n_a$ : integer;

$\mathbf{I}_0$ : $\{n(v_0) = 0$ and no message has arrived at $v_0\}$
begin
    $n(v_0) := 1$;
    $M(v_0) := \{(\lambda(v_0), 1, \emptyset)\}$;
    for $i := 1$ to $deg(v_0)$ do send$< (n(v_0), M(v_0)), i >$ via port $i$ ;
end

$\mathbf{R}_0$ : $\{$A message $< (n_a, M_a)\ , p >$ has arrived at $v_0$ from port $q\}$
begin
    $M := M(v_0)$;
    $M(v_0) := M(v_0) \cup M_a$;
    if $((x, p, q) \notin N(v_0)$ for some $x)$ then
        $N(v_0) := N(v_0) \cup \{(n_a, p, q)\}$;
    if $((x, p, q) \in N(v_0)$ for some $x < n_a)$ then
        $N(v_0) := (N(v_0) \setminus \{(x, p, q)\}) \cup \{(n_a, p, q)\}$;
    if $(n(v_0) = 0)$ or $(n(v_0) > 0$ and there exists $(l, n(v_0), N) \in M(v_0)$ such that $(\lambda(v_0) <_L l)$ or $((\lambda(v_0) = l)$
    and $(N(v_0) \prec N))))$ then
        $n(v_0) := 1 + \max\{n \in \mathbb{N} \mid (l, n, N) \in M(v_0)$ for some $l, N\}$;
        $M(v_0) := M(v_0) \cup \{(\lambda(v_0), n(v_0), N(v_0))\}$;
    if $(M(v_0) \neq M))$ then
        for $(i := 1$ to $deg(v_0))$ do send $< (n(v_0), M(v_0)), i >$ via port $i$;
end
```

An Order on Local Views. We assume for the rest of this paper that the set of labels L is totally ordered by $<_L$. Consider a vertex v such that the local view $N(v) \in \mathcal{N}$ is the set $\{(n_1, p_{s,1}, p_{r,1}), (n_2, p_{s,2}, p_{r,2}), \ldots, (n_d, p_{s,d}, p_{r,d})\}$.

We assume that for each $i < d$, $(n_{i+1}, p_{s,i+1}, p_{r,i+1}) <_{Lex} (n_i, p_{s,i}, p_{r,i})$ where $<_{Lex}$ denotes the usual lexical order. We say that $((n_1, p_{s,1}, p_{r,1}), (n_2, p_{s,2}, p_{r,2}), \ldots, (n_d, p_{s,d}, p_{r,d}))$ is the ordered representation $N_>(v_0)$ of the local view of v_0. Let $\mathcal{N}_>$ be the set of such ordered tuples. We define a total order $\prec$ on $\mathcal{N}_>$ using the alphabetical order that induces naturally a total order on $\mathcal{N}$. This order can also be defined on $\mathcal{N}$ as follows: $N_1 \prec N_2$ if the maximal element for the lexical order $<_{Lex}$ of the symmetric difference $N_1 \bigtriangleup N_2 = N_1 \cup N_2 \setminus N_1 \cap N_2$ belongs to N_2. If $N(u) \prec N(v)$, then we say that the local view $N(v)$ of v is stronger than the one of u.

The Final Labelling. Let $\mathbf{G} = (G, \lambda)$ be a connected labelled graph with the port numbering ν. If v is a vertex of G then the label of v after a run ρ of $\mathcal{M}$ is denoted $(\lambda(v), c_\rho(v))$ with $c_\rho(v) = (n_\rho(v), N_\rho(v), M_\rho(v))$ and (λ, c_ρ) denotes the final labelling. Finally $\mathcal{M}$ verifies:

Proposition 8. *Any run ρ of $\mathcal{M}$ on $\mathbf{G} = (G, \lambda)$, a connected labelled graph with the port numbering ν, terminates and yields a final labelling (λ, c_ρ) verifying the following conditions for all vertices v, v' of G:*

1. *there exists an integer $k \leq V(G)$ such that $\{n_\rho(v) \mid v \in V(G)\} = [1, k]$.*
2. *$M_\rho(v) = M_\rho(v')$.*
3. *$(\lambda(v), n_\rho(v), N_\rho(v)) \in M_\rho(v')$.*
4. *Let $(l, n, N) \in M_\rho(v')$. Then $\lambda(v) = l$, $n_\rho(v) = n$ and $N_\rho(v) = N$ for some vertex v if and only if there is no triple $(l', n, N') \in M_\rho(v')$ with $l <_L l'$ or $(l = l'$ and $N \prec N')$.*
5. *$n_\rho(v) = n_\rho(v')$ implies $(\lambda(v) = \lambda(v')$ and $N(v) = N(v'))$.*

For a mailbox M, we define the graph of the “strongest” vertices as follows. First, for $l \in L, n \in \mathbb{N}, N \in \mathcal{N}, M \subseteq L \times \mathbb{N} \times \mathcal{N}$, we define the predicate $\underline{\text{Strong}}(l, n, N, M)$ that is true if there is no $(l', n, N') \in M$ verifying

$$l < l' \text{ or } (l = l' \text{ and } N \prec N').$$

The digraph H_M of strongest vertices of M is then defined by

$$\begin{aligned}
V(H_M) &= \{n \mid \exists N, l : \underline{\text{Strong}}(l, n, N, M)\},\\
A(H_M) &= \{a = (n, p, q, n') \text{ with } s(a) = n, t(a) = n', Sym(a) = (n', q, p, n)\\
&\qquad \mid \exists N, l \text{ s.t. } \underline{\text{Strong}}(l, n, N, M), \text{ and } \exists p, q \text{ s.t. } (n', p, q) \in N \}.
\end{aligned}$$

We also define a labelling on this digraph by $\lambda_M(n) = l$, with $\underline{\text{Strong}}(n, l, N, M)$ for some N.

The uniqueness of this definition comes from the definition of $\underline{\text{Strong}}$ and from Theorem 8.5.

Let ρ be a run of $\mathcal{M}$. Then $(H_{M_\rho(u)}, \lambda_{M_\rho(u)})$ does not depend on u by Theorem 8.2. We then define $\rho(\mathbf{G}) = (H_{M_\rho(u)}, \lambda_{M_\rho(u)})$, for any vertex u. The labelling defined also a canonical port numbering denoted ν_ρ. Finally, we have:

Proposition 9. *For a given execution ρ of Mazurkiewicz algorithm, we have*

$$V(\rho(\mathbf{G})) = \{n_\rho(v) | v \in V(G)\},$$

$$A(\rho(\mathbf{G})) = \{(n_\rho(v), p, q, n_\rho(w)) | \{v, w\} \in E(G)\},$$

and it is equipped with the port numbering ν_ρ induced by the labelling.

Remark 10. Before we emphasize the role of $\rho(\mathbf{G})$, note that $\rho(\mathbf{G})$ can be locally computed by every vertex, and that the graph depends only on the label M_ρ.

The next proposition states that we can see a run of $\mathcal{M}$ as computing a graph covered by $\widehat{\mathbf{G}}$. Conversely, every graph covered by $\widehat{\mathbf{G}}$ can be obtained by a run of the algorithm.

Proposition 11. *Let $\mathbf{G}$ be a labelled graph equipped with a port numbering ν.*

1. *For all runs ρ of $\mathcal{M}$, $\widehat{\mathbf{G}}$ is a covering of $\widehat{\rho(\mathbf{G})}$.*
2. *(completeness) For all $\mathbf{H}$ such that $\widehat{\mathbf{G}}$ is a covering of $\widehat{\mathbf{H}}$, there exists a run ρ such that $\widehat{\mathbf{H}} \simeq \widehat{\rho(\mathbf{G})}$.*

If the underlying graph is covering-minimal, then $\widehat{\rho(\mathbf{G})}$ is an isomorphic copy of $\widehat{\mathbf{G}}$. This copy can be computed from their mailbox by any vertex, providing a "map" – with numbers of identification – of the underlying network. Thus, on minimal networks, the algorithm of Mazurkiewicz can actually be seen as a *cartography algorithm.*

Interpretation of the Mailboxes at the Step i. The previous results concern the interpretation of the final mailboxes. Now, we consider a relabelling chain $(\mathbf{G}_i)_{0\le i}$. For a given i and a given vertex v we prove that it is possible to interpret the label of v in $\mathbf{G}_i$ as a graph quasi-covered by $\widehat{\mathbf{G}_i}$. We recall notation. Let $\mathbf{G}$ be a labelled graph equipped with a port numbering. Let ρ be a run of the Mazurkiewicz algorithm and let $(\mathbf{G}_i)_{0\le i}$ be a chain associated to ρ with ($\mathbf{G}_0 = \mathbf{G}$). If v is a vertex of $\mathbf{G}$ then the label of v at step i is denoted by $(\lambda(v), c_i(v)) = (\lambda(v), (n_i(v), N_i(v), M_i(v)))$. Using the interpretation of the previous section by defining $\underline{\text{Strong}}(M_i(v))$, this label enables in some cases the reconstruction of the graph $\mathbf{H}_{M_i(v)}$. We note

$$\mathbf{H}_i(v) = \begin{cases} \mathbf{H}_{M_i(v)} \text{if it is defined and } (n_i(v), \lambda(v), N_i(v)) \in \underline{\text{Strong}}(M_i(v)) \\ \perp \text{ otherwise.} \end{cases} \tag{1}$$

We prove that $\widehat{\mathbf{G}_i}$ is a quasi-covering of $\widehat{\mathbf{H}_i(v)}$. First, we need a definition:

Definition 12. *Let $(\mathbf{G}_i)_{0\le i}$, be a relabelling chain obtained with the Mazurkiewicz algorithm and let v be a vertex. We associate to the vertex v and to the step i the integer $r^{(i)}_{agree}(v)$ being the maximal integer bounded by the diameter of $\mathbf{G}$ such that any vertex w of $B(v, r^{(i)}_{agree}(v))$ verifies: $\mathbf{H}_i(v) = \mathbf{H}_i(w)$.*

Now we can state the main result of this section:

Theorem 13. *Let $(\mathbf{G}_i)_{0\le i}$, be a relabelling chain obtained with the Mazurkiewicz algorithm and let v be a vertex. The graph $\widehat{\mathbf{G}_i}$ is a quasi-covering of $\widehat{\mathbf{H}_i(v)}$ centered on v of radius $r^{(i)}_{agree}(v)$.*

5 An Algorithm to Detect Stable Properties

In this section we describe a generalisation of the algorithm by Szymanski, Shy and Prywes (the SSP algorithm for short) [SSP85]. We consider a distributed algorithm which terminates when all processes reach their local termination conditions. Each process is able to determine only its own termination condition. The SSP algorithm detects an instant in which the entire computation is achieved.

We present here a generalization of the hypothesis under which the SSP rules are run. For every vertex v, the value of $P(v)$ is no more a boolean and can have any value which depends on the label (state) of v denoted by $\texttt{state}(v)$. Hence, we *do not* require each process to determine when it reachs its own termination condition. Moreover the function P must verify the following property: for any α, if $P(\texttt{state}(v))$ has the value α ($\alpha \neq \perp$) and changes to $\alpha' \neq \alpha$ then it can not be equal to α at an other time. In other words, under this hypothesis, the function is constant between two moments where it has the same value (different from $\perp$). We say that the function P is *value-convex*. We extend the SSP rules and we shall denote by GSSP this generalisation. In GSSP, the counter of v is incremented only if P is constant on the ball $B(v)$. As previously, every underlying rule that computes in particular $P(\texttt{state}(v))$, has to be modified in order to eventually reinitialize the counter. Initially $a(v) = -1$ for all vertices. The GSSP rule modifies the counter a.

Algorithm 2: Algorithm GSSP.

```
Var : $a(v_0)$ : integer init $-1$ ;
      $t_{v_0}[i]$ : integer init $-1$ for each port $i$ of $v_0$;
      $val_{v_0}[i]$ : value init $\perp$ for each port $i$ of $v_0$;
      $i, j, x, temp$ : integer;

C$_0$ : {A new value $P(\texttt{state}(v_0)) = \perp$ is computed}
begin
    $a(v_0) := -1$ ;
    for $i := 1$ to $deg(v_0)$ do send$< \perp, -1 >$ via port $i$ ;
end

C$_1$ : {A new value $P(\texttt{state}(v_0))$ different from $\perp$ is computed}
begin
    $a(v_0) := 0$ ;
    if ($P(\texttt{state}(v_0))$ is equal to $val_{v_0}[i]$ for each port $i$) then
        $a(v_0) := 1$;
    for $i := 1$ to $deg(v_0)$ do send$< P(\texttt{state}(v_0)), a(v_0) >$ via port $i$ ;
end

C$_2$ : {A message $< \alpha, x >$ has arrived at $v_0$ from port $j$}
begin
    $val_{v_0}[j] := \alpha$;
    $t_{v_0}[j] := x$;
    $temp := a(v_0)$;
    if ($P(\texttt{state}(v_0)) \neq \perp$ is equal to $val_{v_0}[i]$ for each port $i$) then
        $a(v_0) := 1 + Min\{t_{v_0}[i] \mid i \text{ is a port of } v_0\}$;
    if ($temp \neq a(v_0)$) then
        for $i := 1$ to $deg(v_0)$ do send$< P(\texttt{state}(v_0)), a(v_0) >$ via port $i$ ;
end
```

We shall now use the following notation. Let $\mathbf{G}$ be a labelled graph equipped with a port numbering ν. Let $(\mathbf{G}_i)_{0 \leq i}$ be a relabelling chain associated to an execution of the GSSP algorithm. We denote by $a_i(v)$ (resp. $P_i(\texttt{state}(v))$) the value of the counter (resp. of the function) associated to the vertex v of $\mathbf{G}_i$. According to the definition of the GSSP rule, we remark that for every vertex v, $a(v)$ can be increased, at each step, by 1 at most and that if $a(v)$ increases from h to $h+1$, that means that at the previous step, all the neighbours w of v were such that $a(w) \geq h$ and $P(\texttt{state}(w)) = P(\texttt{state}(v))$. Now we give the fundamental property of GSSP algorithm.

Lemma 14 (GSSP). *Consider an execution of the GSSP algorithm under the hypothesis that the function P is value-convex. For all j, for all v, there exists $i \leq j$ such that for all $w \in B(v, \lfloor \frac{a_j(v)}{3} \rfloor)$, $P_i(\texttt{state}(w)) = P_j(\texttt{state}(v))$.*

5.1 Mazurkiewicz Algorithm + GSSP algorithm = Maximal Common Knowledge

The main idea in this section is to use the GSSP algorithm in order to compute, in each node, the radius of stability of $\mathcal{M}$. In other words, each node u will know how far other nodes agree with its reconstructed graph

$\mathbf{H}_{M(u)}$. Let $\mathbf{G} = (G, \lambda)$ be a labelled graph equipped with a port numbering, let $(\mathbf{G}_i)_{0 \leq i}$ be a relabelling chain associated to a run of Mazurkiewicz' Algorithm on the graph $\mathbf{G}$. The vertex v of $\mathbf{G}_i$ is associated to the label $(\lambda(v), (n_i(v), N_i(v), M_i(v)))$. Using the interpretation of Section 4, this labelling enables to compute the maximal common knowledge for all vertices.

Let's consider the algorithm obtained by adding to each rule of the Mazurkiewicz algorithm, the calculus of $\mathbf{H}_i(v)$ on each node v and the modifications for the GSSP rule.

We note $\mathcal{AS}$ the merging of the two algorithms. The output of $\mathcal{AS}$ on the node v is $< \mathbf{H}_i(v), a_i(v) >$.

¿From Lemma 14 and Theorem 13, the main property of the computation of $\mathcal{AS}$ is:

Theorem 15 (quasi-covering progression). *At all step j, for all vertex v, the output of $\mathcal{AS}$ on v is a couple $< \mathbf{H}_j(v), a_j(v) >$ such that if $\mathbf{H}_j \neq \bot$, then there exists a previous step $i < j$, such that $\widehat{\mathbf{G}}_i$ is a quasi-covering of $\widehat{\mathbf{H}_i(v)}$ of center v and of radius $\lfloor \frac{a_j(v)}{3} \rfloor$.*

And as the underlying Mazurkiewicz Algorithm is always terminating, we have that the value of $\mathbf{H}$ will stabilize with a going to the infinite.

Finally, and considering the previous theorem, we note $r^{\mathrm{t}} = \lfloor \frac{a_j(v)}{3} \rfloor$, the *radius of trust* for the algorithm $\mathcal{AS}$.

6 Termination Detection

6.1 Some Definitions

As it is said in Remark 7 distributed algorithm is synonymous with local computations on arcs.

First, we recall from the previous section: let $\mathcal{R}$ be a locally generated relabelling relation, let $\mathbf{D}$ a labelled digraph, we say that $\mathbf{D}$ is an irreducible configuration modulo $\mathcal{R}$ if $\mathbf{D}$ is a $\mathcal{R}$-normal form, i.e., no further step with $\mathcal{R}$ is possible ($\mathbf{D}\mathcal{R}\mathbf{D}'$ holds for no $\mathbf{D}'$).

Irreducibility with respect to a relabelling relation yields a notion of implicit termination : the computation has ended – no more relabelling rule can be applied – but no node is aware of the termination. On the other hand, one shall ask a node to be aware of the termination of the algorithm. We consider two kinds of terminations :

– Termination of the algorithm but without detection : *implicit termination.*
– The nodes know when all other nodes have computed their final output value. Due to the asynchronous aspect of distributed computations, there is still some *observational* computations that are going on. This is the *observed termination detection* as when termination is detected, some observation computations are not necessarily terminated; it is called usually explicit termination.

We give the formal definition of the observed termination detection for digraphs relabelling systems. In order to have a unified presentation, we restrict ourselves to "normalized relabelling systems" w.l.o.g.

Definition 16. *A* normalized labelled digraph $\mathbf{D}$ *is a labelled digraph whose labelling is of the form* ($\mathtt{mem}$, $\mathtt{out}$, $\mathtt{term}$).

A normalized relabelling system $\mathcal{R}$ *is a digraph relabelling system on normalized digraphs where : $\mathtt{mem}$ can be used in preconditions and relabelled, $\mathtt{out}$ is only relabelled, $\mathtt{term}$ is only relabelled and has a value in $\{\bot, \text{TERM}\}$. We also use the following convention : if the initial labelled digraph is $\mathbf{D} = (D, in)$ then it is implicitly extended to the normalized labelling $(D, (in, \bot, \bot))$. The initial value of $\mathtt{mem}$ is therefore given by in.*

All digraphs are labelled digraphs and are now all considered to be normalized. All relabelling relations are relabelling relations of normalized labelled digraphs. We also use the following notations. Let $\mathbf{D}$ and $\mathbf{D}$' be some given normalized digraphs then, for any vertex $u \in \mathbf{D}$ (resp. $\in \mathbf{D}'$), for any $\mathtt{x} \in \{\mathtt{mem}, \mathtt{out}, \mathtt{term}\}$, $\mathtt{x}(u)$ (resp. $\mathtt{x}'(u)$) is the $\mathtt{x}$ component of u in $\mathbf{D}$ (resp. $\mathbf{D}$'). This presentation will find its justifications with the following definitions.

For the implicit termination, there is no detection mechanism. Hence $\mathtt{term}$ is not used. If the underlying distributed algorithm is aimed at the computation of a special value, we will, in order to distinguish this value from the intermediate computed values, only look the special purpose component $\mathtt{out}$. As there is no detection of termination, this label is written all over the computation. It becomes significant only when the digraph is irreducible, but no node knows when this happens.

Now we give the definition of the observed termination detection.

Definition 17. *Let $\mathcal{F}$ be a family of labelled digraphs. A digraph relabelling relation $\mathcal{R}$ has an* observed termination detection *(OTD) on $\mathcal{F}$ if :*

17.i $\mathcal{R}$ is noetherian on $\mathcal{F}$,
17.ii the $\mathtt{term}$ component of $\mathcal{R}$-irreducible digraphs is equal to TERM,
17.iii for all digraphs $\mathbf{D}$, $\mathbf{D}' \in \mathcal{F}$ such that $\mathbf{D}\mathcal{R}^*\mathbf{D}'$, if there exists a vertex u such that $\mathtt{term}(u) = \text{TERM}$, then

– $\mathtt{term}'(u) = \textsc{Term}$,
– *for all vertex* $v \in \mathbf{D}$, $\mathtt{out}'(v) = \mathtt{out}(v)$.

In this definition, we ask the network to detect the termination of the computation (in the sense of the `out` value that is computed), but not to detect the termination of that detection. We have at least one vertex that detects that the `out` values are final and then it can perform a broadcast of TERM. This broadcast is performed by an "observer algorithm" whose termination we do not consider.

6.2 Some Useful Properties

Definition 18. *Let $\mathcal{F}$ be a digraph family. We denote by $\mathcal{F}\!\downarrow$ the family of digraphs that are covered by a digraph of $\mathcal{F}$: $\mathcal{F}\!\downarrow = \{\mathbf{D}' \mid \exists \mathbf{D} \in \mathcal{F}, \mathbf{D}$ is a covering of $\mathbf{D}'\}$.*

Note that $\mathcal{F}$ is a subset of $\mathcal{F}\!\downarrow$. We have:

Lemma 19. *Let $\mathcal{R}$ be a relabelling system. If $\mathcal{R}$ is noetherian on $\mathcal{F}$, it is also noetherian on $\mathcal{F}\!\downarrow$.*

We now present the fundamental lemma, due to Angluin [Ang80], connecting coverings and locally generated relabelling relations. It states that whenever $\mathbf{D}$ is a covering of $\mathbf{D}'$, every relabelling step in $\mathbf{D}'$ can be lifted to a relabelling sequence in $\mathbf{D}$, which is compatible with the covering relation.

Lemma 20 (Lifting Lemma). *Let $\mathcal{R}$ be a locally generated relabelling relation and let $\mathbf{D}$ be a covering of $\mathbf{D}'$ via γ. If $\mathbf{D}'\ \mathcal{R}^*\ \mathbf{D}'_1$ then there exists $\mathbf{D}_1$ such that $\mathbf{D}\ \mathcal{R}^*\ \mathbf{D}_1$ and $\mathbf{D}_1$ is a covering of $\mathbf{D}'_1$ via γ.*

Quasi-coverings have been introduced to study the problem of the detection of the termination in [MMW97]. The idea behind them is to enable the partial simulation of local computations on a given digraph in a restricted area of a larger digraph. The restricted area where we can perform the simulation will shrink while the number of simulated steps increases. The following lemma makes precise how much the radius shrinks when one step of simulation is performed :

Lemma 21 (Quasi-Lifting Lemma). *Let $\mathcal{R}$ be a locally generated relabelling relation and let $\mathbf{D}$ be a quasi-covering of $\mathbf{D}'$ of radius r via γ. Moreover, let $\mathbf{D}'\ \mathcal{R}\ \mathbf{D}'_1$. Then there exists $\mathbf{D}_1$ such that $\mathbf{D}\ \mathcal{R}^*\ \mathbf{D}_1$ and $\mathbf{D}_1$ is a quasi-covering of radius $r-2$ of $\mathbf{D}'_1$.*

6.3 The Main Result

Let $\mathcal{G}$ be a recursive family of labelled graphs equipped with a port numbering. Let $\mathcal{F}$ be the family of labelled digraphs obtained from $\mathcal{G}$ and defined by: $\mathcal{F} = \{\mathbf{D} \mid \exists \mathbf{G} \in \mathcal{G}$ and $\mathbf{D} = \widehat{\mathbf{G}}\}$. Let $\mathcal{R}$ be a noetherian digraph relabelling relation on arcs, now we can state the characterization for the existence of an equivalent relation with the observed termination detection. With the notation of this section:

Theorem 22. *For family $\mathcal{F}$, there exists a transformation that maps any noetherian digraph relabelling relation on arcs $\mathcal{R}$ to a noetherian digraph relabelling relation on arcs with observed termination detection if and only if there exists a recursive function $r : \mathcal{F}\!\downarrow \longrightarrow \mathbb{N}$ such that for any $\mathbf{D}' \in \mathcal{F}\!\downarrow$, there is no strict quasi-covering of $\mathbf{D}$' of radius $r(\mathbf{D}')$ in $\mathcal{F}$.*

Proof. **Necessary Condition.** This is actually a simple corollary of the quasi-lifting lemma. We prove this by contradiction. We suppose there exists $\mathbf{D}' \in \mathcal{F}\!\downarrow$ that admits strict quasi-coverings of unbounded radius in $\mathcal{F}$. By Lemma 19, $\mathcal{R}$ is noetherian for $\mathbf{D}$'. Consider an execution of $\mathcal{R}$ of length l. By hypothesis, there exists $\mathbf{D}'' \in \mathcal{F}$ a strict quasi-covering of $\mathbf{D}$' of radius $2l+1$. By the quasi-lifting lemma, we can simulate on a ball of radius $2l+1$ of $\mathbf{D}$" the execution of $\mathcal{R}$ on $\mathbf{D}$'. At the end of this relabelling steps, there is a node in $\mathbf{D}$" that is labelled TERM. As the quasi-covering $\mathbf{D}$" is strict, there exists at least one node outside of the ball that has not even taken a relabelling step of $\mathcal{R}$, hence that has not written anything to `out`. Hence $\mathcal{R}$ has not the observed termination property on $\mathbf{D}$". A contradiction.

Sufficient Condition. The main idea is to compose $\mathcal{R}$ with the algorithm $\mathcal{AS}$. On each vertex v of D and for each port i of v we define two counters $c_{out}(i)$ and $c_{in}(i)$: $c_{out}(i)$ stores the number of basic messages sent by v via i for $\mathcal{R}$ and $c_{in}(i)$ stores the number of messages received by v via i for $\mathcal{R}$.

We use the algorithm $\mathcal{AS}$ of Subsection 5.1 with notation of digraphs (we use $\mathbf{D}'$ instead of $\mathbf{H}$). We add the values of the counters of the vertex to the messages that GSSP sends. We add to $\mathcal{AS}$ for each vertex v and for each port i the number of basic messages in the corresponding channel that we deduce from the values of the counters.

Now we consider the following termination detection condition: each channel is empty, and $\mathbf{D}$' is irreducible for $\mathcal{R}$, and there exists $\mathbf{D} \in \mathcal{F}$ such that $\mathbf{D}$ is a covering of $\mathbf{D}$', and $r(\mathbf{D}'(v)) < r^t(v)$. To test if there exists $\mathbf{D} \in \mathcal{F}$ such that $\mathbf{D}$ is a covering of $\mathbf{D}$', we enumerate always in the same order all the graphs of $\mathcal{F}$ by order of increasing diameter. We denote $\mathrm{AS}_{\mathcal{R}}$ this algorithm. If $\mathcal{R}$ is noetherian then $\mathrm{AS}_{\mathcal{R}}$ is noetherian: as $\mathcal{R}$ is noetherian this implies that the numbers of input-values for computing $\mathbf{D}$' is bounded and the result follows.

Known Results as Corollaries. This theorem admits well known corollaries; more precisely we deduce immediatly that in the asynchronous message passing model a distributed algorithm having an implicit termination may be transformed into a distributed algorithm having an observed (explicit) termination detection for the following families of graphs : graphs having a distinguished vertex, graphs such that each node is identified by a unique name graphs having a known size or diameter bounds the family of connected subgraphs of grids with a sense of direction trees. We deduce there is no observed (explicit) termination detection for : the family of rings, the family of connected subgraphs of grids without sense of direction, the family of rings having a prime size.
New Corollaries. New corollaries are obtained from this theorem; in the asynchronous message passing model a distributed algorithm having an implicit termination may be transformed into a distributed algorithm having an observed (explicit) termination detection for the following families of graphs : graphs having exactly k leaders (distinguished vertices), graphs having at least one and at most k, leaders (distinguished vertices).

For the election problem this theorem and results of [CM05] imply:

Theorem 23. *For family $\mathcal{F}$, there exists an election algorithm if and only if graphs of $\mathcal{F}$ are minimal for the covering relation and there exists a recursive function $r : \mathcal{F} \longrightarrow \mathbb{N}$ such that for any $\mathbf{D} \in \mathcal{F}$, there is no strict quasi-covering of $\mathbf{D}$ of radius $r(\mathbf{D})$ in $\mathcal{F}$, except $\mathbf{D}$ itself.*

References

[Ang80] D. Angluin. Local and global properties in networks of processors. In *Proc. of the 12th Symposium on Theory of Computing (STOC 1980)*, pages 82–93, 1980.

[AW04] H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics.* John Wiley and Sons, 2004.

[BCG$^+$96] P. Boldi, B. Codenotti, P. Gemmell, S. Shammah, J. Simon, and S. Vigna. Symmetry breaking in anonymous networks: characterizations. In *Proc. of the 4th Israeli Symposium on Theory of Computing and Systems (ISTCS 1996)*, pages 16–26. IEEE Press, 1996.

[Bod89] H.L. Bodlaender. The classification of coverings of processor networks. *Journal of parallel and distributed computing*, 6(1):166–182, 1989.

[BV99] P. Boldi and S. Vigna. Computing anonymously with arbitrary knowledge. In *Proc. of the 18th ACM Symposium on principles of distributed computing (PODC 1999)*, pages 181–188. ACM Press, 1999.

[BV01] P. Boldi and S. Vigna. An effective characterization of computability in anonymous networks. In *Proc. of Distributed Computing, 15th International Conference (DISC 2001)*, volume 2180 of *Lecture Notes in Computer Science*, pages 33–47. Springer-Verlag, 2001.

[BV02] P. Boldi and S. Vigna. Fibrations of graphs. *Discrete Mathematics*, 243(1-3):21–66, 2002.

[BvL86] H.L. Bodlaender and J. van Leeuwen. Simulation of large networks on smaller networks. *Information and Control*, 71(3):143–180, 1986.

[CM05] J. Chalopin and Y. Métivier. A bridge between the asynchronous message passing model and local computations in graphs. In *Proc. of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS 2005)*, volume 3618 of *Lecture Notes in Computer Science*, pages 212–223. Springer-Verlag, 2005.

[IR81] A. Itai and M. Rodeh. Symmetry breaking in distributive networks. In *Proc. of the 22nd Annual Symposium on Foundations of Computer Science (FOCS 1981)*, pages 150–158. IEEE, 1981.

[Mat87] F. Mattern. Algorithms for distributed termination detection. *Distributed computing*, 2(3):161–175, 1987.

[Maz97] A. Mazurkiewicz. Distributed enumeration. *Information Processing Letters*, 61(5):233–239, 1997.

[MMW97] Y. Métivier, A. Muscholl, and P.-A. Wacrenier. About the local detection of termination of local computations in graphs. In *Proc. of the 4th International Colloquium on Structural Information and Communication Complexity (SIROCCO 1997)*, Proceedings in Informatics, pages 188–200. Carleton Scientific, 1997.

[MT00] Y. Métivier and G. Tel. Termination detection and universal graph reconstruction. In *Proc. of 7th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2000)*, Proceedings in Informatics, pages 237–251. Carleton Scientific, 2000.

[SSP85] B. Szymanski, Y. Shy, and N. Prywes. Synchronized distributed termination. *IEEE Transactions on software engineering*, 11(10):1136–1140, 1985.

[Tel00] G. Tel. *Introduction to distributed algorithms.* Cambridge University Press, 2000.

[YK96] M. Yamashita and T. Kameda. Computing on anonymous networks: Part i - characterizing the solvable cases. *IEEE Transactions on parallel and distributed systems*, 7(1):69–89, 1996.